


```
LL      IIIIII  BBBB BBBB  AAAAAA  DDDDDDDD  DDDDDDDD  XX      XX
LL      IIIIII  BBBB BBBB  AAAAAA  DDDDDDDD  DDDDDDDD  XX      XX
LL      II      BB      BB  AA      AA  DD      DD  DD      DD  XX      XX
LL      II      BB      BB  AA      AA  DD      DD  DD      DD  XX      XX
LL      II      BB      BB  AA      AA  DD      DD  DD      DD  XX      XX
LL      II      BBBB BBBB  AA      AA  DD      DD  DD      DD  XX      XX
LL      II      BBBB BBBB  AA      AA  DD      DD  DD      DD  XX      XX
LL      II      BB      BB  AAAAAAAAAA  DD      DD  DD      DD  XX      XX
LL      II      BB      BB  AAAAAAAAAA  DD      DD  DD      DD  XX      XX
LL      II      BB      BB  AA      AA  DD      DD  DD      DD  XX      XX
LL      II      BB      BB  AA      AA  DD      DD  DD      DD  XX      XX
LLLLLLLLLLLL  IIIIII  BBBB BBBB  AA      AA  DDDDDDDD  DDDDDDDD  XX      XX
LLLLLLLLLLLL  IIIIII  BBBB BBBB  AA      AA  DDDDDDDD  DDDDDDDD  XX      XX
                                         ....
                                         ....
                                         ....
                                         ....
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
```


(2)	56
(3)	98
(4)	215

DECLARATIONS
LIB\$ADDX - Addition of infinite precision integers
LIB\$SUBX - Subtraction of infinite precision integers

```
0000 1 .TITLE LIB$ADDX - Add infinite precision integers
0000 2 .IDENT /1-006/ ; File: LIBADDX.MAR Edit: PDG1006
0000 3
0000 4 :
0000 5 :*****
0000 6 :*
0000 7 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :* ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :* TRANSFERRED.
0000 17 :*
0000 18 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :* CORPORATION.
0000 21 :*
0000 22 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28 :
0000 29 :++
0000 30 : FACILITY: General Utility Library
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 : Routines for performing addition and subtraction on
0000 35 : integers of arbitrary length.
0000 36 :
0000 37 : ENVIRONMENT: User Mode, AST Reentrant
0000 38 :
0000 39 :--
0000 40 : AUTHOR: Steven B. Lionel, CREATION DATE: 17-NOV-1978
0000 41 :
0000 42 : MODIFIED BY:
0000 43 :
0000 44 : Steven B. Lionel, : VERSION 01
0000 45 : 1-001 - Original
0000 46 : 1-002 - Corrected an error in a comment. JBS 14-DEC-78
0000 47 : 1-003 - Add "" to PSECT directive. JBS 21-DEC-78
0000 48 : 1-004 - Make default loop count 1 instead of 2. A value of 2
0000 49 : makes it loop 3 times! SBL 25-MAR-1980
0000 50 : 1-005 - Use register temp in loop to allow for overlap. SBL 13-June-1980
0000 51 : 1-006 - Allow length of zero or one. Recognize overflow. Recognize length
0000 52 : of -2**31 as an error. Made compare with (AP) unsigned. Make sure
0000 53 : that C bit is 0 before entering loop. PDG 9-Aug-81
0000 54 :--
```



```
0000 56 .SBTTL DECLARATIONS
0000 57 :
0000 58 : INCLUDE FILES:
0000 59 :
0000 60 :
0000 61 :
0000 62 : EXTERNAL DECLARATIONS:
0000 63 :
0000 64 .DSABL GBL ; Prevent undeclared symbols
0000 65 ; from being declared global
0000 66
0000 67 .EXTRN SSS_NORMAL ; Normal successful completion
0000 68 .EXTRN SSS_INTOVF ; Integer overflow error
0000 69 .EXTRN LIB$_INVARG ; Invalid argument to function
0000 70 :
0000 71 : MACROS:
0000 72 :
0000 73 :
0000 74 :
0000 75 : EQUATED SYMBOLS:
0000 76 :
00000004 0000 77 addend = 4 ; Address of addend array
00000008 0000 78 augend = 8 ; Address of augend array
0000000C 0000 79 sum = 12 ; Address of sum (result) array
0000 80
00000004 0000 81 minuend = 4 ; Address of minuend array
00000008 0000 82 subtrahend = 8 ; Address of subtrahend array
0000000C 0000 83 difference = 12 ; Address of difference array
0000 84
00000010 0000 85 length = 16 ; Address of length in longwords
0000 86
0000 87 :
0000 88 : OWN STORAGE:
0000 89 :
0000 90 :
0000 91 :
0000 92 : PSECT DECLARATIONS:
0000 93 :
00000000 0000 94 .PSECT _LIB$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 95 EXE, RD, NOWRT, LONG
0000 96
```

```
0000 98 .SBTTL LIB$ADDX - Addition of infinite precision integers
0000 99 :++
0000 100 : FUNCTIONAL DESCRIPTION:
0000 101 :
0000 102 : LIB$ADDX performs addition of arbitrary length integers. The
0000 103 : values to be added are located in arrays of longwords: the
0000 104 : higher addresses holding the more significant parts of the values.
0000 105 :
0000 106 : The number of longwords to be added is given in the optional
0000 107 : argument "length". If this is not specified, the default is
0000 108 : 2, or quadword addition.
0000 109 :
0000 110 : The sum is placed in the array addressed by the third argument.
0000 111 : Any two or all three of the first three arguments may be the same.
0000 112 : If overflow occurs, the function value returned is $$$_INTOVF.
0000 113 :
0000 114 : CALLING SEQUENCE:
0000 115 :
0000 116 : status.wlc.v = LIB$ADDX (addend.rl.ra, augend.rl.ra, sum.wl.ra
0000 117 : [, length.rl])
0000 118 :
0000 119 : INPUT PARAMETERS:
0000 120 :
0000 121 : addend - The address of an array of longwords. The array
0000 122 : contains a multiple precision integer, with the
0000 123 : bits increasing in significance with increasing
0000 124 : addresses.
0000 125 :
0000 126 : augend - The address of an array of longwords. The array
0000 127 : contains a multiple precision integer, with the
0000 128 : bits increasing in significance with increasing
0000 129 : addresses.
0000 130 :
0000 131 : length - Optional. The length in longwords of the arrays to
0000 132 : be added. The length may be zero or greater. If
0000 133 : not, error LIB$_INVARG is returned.
0000 134 :
0000 135 : IMPLICIT INPUTS:
0000 136 :
0000 137 : NONE
0000 138 :
0000 139 : OUTPUT PARAMETERS:
0000 140 :
0000 141 : sum - The address of an array of longwords. The sum of the
0000 142 : addend and augend is placed in this array.
0000 143 :
0000 144 : IMPLICIT OUTPUTS:
0000 145 :
0000 146 : NONE
0000 147 :
0000 148 : FUNCTION VALUE:
0000 149 : COMPLETION CODES:
0000 150 :
0000 151 : $$$_NORMAL - Successful completion
0000 152 : $$$_INTOVF - Integer overflow - sum is correct except for
0000 153 : the sign bit which is lost.
0000 154 : LIB$_INVARG - Invalid argument. Length is negative.
```



```
0000 155 : The sum is not changed.
0000 156 :
0000 157 : SIDE EFFECTS:
0000 158 :
0000 159 : NONE
0000 160 :
0000 161 :--
0000 162 :
0004 0000 163 .ENTRY LIB$ADDX, ^M<R2> ; Disable integer overflow
0002 164 :
0002 165 :+
0002 166 :
0002 167 : Set up R0 as the count of longwords remaining. If length
0002 168 : is not specified, use the default of 2 (quadword addition).
0002 169 : If length is negative, return error LIB$_INVARG.
0002 170 :-
50 01 7D 0002 171 MOVQ #1, R0 ; Default two longwords
51 D7 0005 172 DECL R1 ; Initialize index
6C 04 91 0007 173 CMPB #4, (AP) ; Is length present?
28 1A 000A 174 BGTRU 2$ ; No, use default (C=0 if branch taken)
10 AC D5 000C 175 TSTL length(AP) ; Is length omitted? (C=0)
23 13 000F 176 BEQL 2$ ; Yes, use default
50 10 BC D0 0011 177 MOVL @length(AP), R0 ; R0 contains length (C=0 (unchanged))
04 15 0015 178 BLEQ 0$ ; Branch if negative or zero length
50 D7 0017 179 DECL R0 ; Subtract one (C=0, since R0 was > 0)
19 11 0019 180 BRB 2$ ; Jump into loop
001B 181 :
001B 182 : User gave negative or zero length
001B 183 :
50 00000000'8F 2E 13 001B 184 0$: BEQL EXIT ; Do nothing if length = 0
D0 001D 185 MOVL #LIB$_INVARG, R0 ; Error
04 0024 186 RET ; Return to caller
0025 187 :
0025 188 :+
0025 189 :
0025 190 :-
0025 191 :
52 04 BC41 D0 0025 192 1$: MOVL @addend(AP)[R1], R2 ; Do addition in a temp because augend
52 08 BC41 D8 002A 193 ADWC @augend(AP)[R1], R2 ; and sum may overlap and because ADWC
OC BC41 52 D0 002F 194 MOVL R2, @sum(AP)[R1] ; is a two-operand instruction
ED 51 50 F2 0034 195 2$: AOBLSS R0, R1, 1$ ; Loop till done
0038 196 :
0038 197 :+
0038 198 :
0038 199 :-
Now, add one more time, this time preserving the overflow flag
OC 52 08 BC41 D0 0038 200 MOVL @augend(AP)[R1], R2
OC BC41 04 BC41 D0 003D 201 MOVL @addend(AP)[R1], @sum(AP)[R1]
OC BC41 52 D8 0044 202 ADWC R2, @sum(AP)[R1]
0049 203 :
0049 204 :+
0049 205 :
0049 206 :-
Test for overflow and return with proper condition
0049 207 :
50 05 1D 0049 208 ATEST: BVS AOVFL ; Integer overflow
9A 004B 209 EXIT: MOVZBL #SS$_NORMAL, R0 ; Return success
04 004F 210 RET
0050 211
```

LIB\$ADDX
1-006

- Add infinite precision integers L 7
LIB\$ADDX - Addition of infinite precisio 15-SEP-1984 23:46:37 VAX/VMS Macro V04-00
6-SEP-1984 11:03:09 [LIBRTL.SRC]LIBADDX.MAR;1

Page 5
(3)

50 0000'8F 3C 0050 212 AOVFL: MOVZWL #SS\$_INTOVF, P.0 ; Integer overflow
04 0055 213 RET ; Return failure


```
0056 215 .SBTTL LIB$SUBX - Subtraction of infinite precision integers
0056 216 :++
0056 217 : FUNCTIONAL DESCRIPTION:
0056 218 :
0056 219 : LIB$SUBX performs subtraction of arbitrary length integers. The
0056 220 : values to be subtracted are located in arrays of longwords: the
0056 221 : higher addresses being the higher precision parts of the values.
0056 222 :
0056 223 : The number of longwords to be subtracted is given in the optional
0056 224 : argument "length". If this is not specified, the default is
0056 225 : 2, or quadword subtraction.
0056 226 :
0056 227 : The difference is placed in the array addressed by the third argument.
0056 228 : Any two or all three of the first three arguments may be the same.
0056 229 : If overflow occurs, the function value returned is $$$_INTOVF.
0056 230 :
0056 231 : CALLING SEQUENCE:
0056 232 :
0056 233 : status.wlc.v = LIB$SUBX (minuend.rl.ra, subtrahend.rl.ra,
0056 234 : difference.wl.ra [, length.rl])
0056 235 :
0056 236 : INPUT PARAMETERS:
0056 237 :
0056 238 : minuend - The address of an array of longwords. The array
0056 239 : contains a multiple precision integer, with the
0056 240 : bits increasing in significance with increasing
0056 241 : addresses.
0056 242 :
0056 243 : subtrahend - The address of an array of longwords. The array
0056 244 : contains a multiple precision integer, with the
0056 245 : bits increasing in significance with increasing
0056 246 : addresses.
0056 247 :
0056 248 : length - Optional. The length in longwords of the arrays to
0056 249 : be subtracted. The length must be greater than one. If
0056 250 : not, error LIB$_INVARG is returned.
0056 251 :
0056 252 : IMPLICIT INPUTS:
0056 253 :
0056 254 : NONE
0056 255 :
0056 256 : OUTPUT PARAMETERS:
0056 257 :
0056 258 : difference - The address of an array of longwords. The subtrahend is
0056 259 : subtracted from the minuend, and the result is placed
0056 260 : in this array.
0056 261 :
0056 262 : IMPLICIT OUTPUTS:
0056 263 :
0056 264 : NONE
0056 265 :
0056 266 : FUNCTION VALUE:
0056 267 : COMPLETION CODES:
0056 268 :
0056 269 : $$$_NORMAL - Successful completion
0056 270 : $$$_INTOVF - Integer overflow - difference is correct except for
0056 271 : the sign bit which is lost.
```

```
0056 272 : LIB$_INVARG - Invalid argument. Length is negative.
0056 273 : The difference is not changed.
0056 274 :
0056 275 : SIDE EFFECTS:
0056 276 :
0056 277 : NONE
0056 278 :
0056 279 :--
0004 0056 280 :
0056 281 : .ENTRY LIB$SUBX, ^M<R2> ; Disable integer overflow
0058 282 :
0058 283 :+
0058 284 : Set up R0 as the count of longwords remaining. If length
0058 285 : is not specified, use the default of 2 (quadword subtraction).
0058 286 : If length is negative, return error LIB$_INVARG.
0058 287 :-
0058 288 :
50 01 7D 0058 289 MOVQ #1, R0 ; Default two longwords
51 D7 005B 290 DECL R1 ; Initialize index
6C 04 91 005D 291 CMPB #4, (AP) ; Is length present?
28 1A 0060 292 BGTRU 2$ ; No, use default (C=0 if branch taken)
10 AC D5 0062 293 TSTL length(AP) ; Is length omitted? (C=0)
23 13 0065 294 BEQL 2$ ; Yes, use default
50 10 BC D0 0067 295 MOVL @length(AP), R0 ; R0 contains length (C=0 (unchanged))
04 15 006B 296 BLEQ 0$ ; Branch if negative or zero length
50 D7 006D 297 DECL R0 ; Subtract one (C=0, since R0 was > 0)
19 11 006F 298 BRB 2$ ; Jump into loop
0071 299 :
0071 300 : User gave negative or zero length
0071 301 :
0071 302 0$: BEQL EXIT ; Do nothing if length = 0
50 00000000'D8 13 0073 303 MOVL #LIB$_INVARG, R0 ; Error
04 007A 304 RET ; Return to caller
007B 305 :
007B 306 :+
007B 307 : Subtraction loop
007B 308 :-
007B 309 :
52 04 BC41 D0 007B 310 1$: MOVL @minuend(AP)[R1], R2 ; Do subtraction in a temp
0080 311 ; because subtrahend and
52 08 BC41 D9 0080 312 SBWC @subtrahend(AP)[R1], R2 ; difference may overlap and
0C BC41 52 D0 0085 313 MOVL R2, @difference(AP)[R1] ; because SBWC is a 2 operand
ED 51 50 F2 008A 314 ; instruction
008A 315 2$: AOBLS R0, R1, 1$ ; Loop till done
008E 316 :
008E 317 :+
008E 318 : Now subtract one final time, this time preserving overflow
008E 319 :-
008E 320 :
0C 52 08 BC41 D0 008E 321 MOVL @subtrahend(AP)[R1], R2
0C BC41 04 BC41 D0 0093 322 MOVL @minuend(AP)[R1], @difference(AP)[R1]
0C BC41 52 D9 009A 323 SBWC R2, @difference(AP)[R1]
009F 324 :
009F 325 :+
009F 326 : Test for overflow and return with proper condition
009F 327 :-
AF 1D 009F 328 BVS AOVFL ; Test for overflow
```


LIB\$ADDX
1-006

- Add infinite precision integers B 8
LIB\$SUBX - Subtraction of infinite preci 15-SEP-1984 23:46:37 VAX/VMS Macro V04-00
6-SEP-1984 11:03:08 [LIBRTL.SRC]LIBADDX.MAR;1

Page 8
(4)

```
50 00'8F 9A 00A1 329  
04 00A1 330 MOVZBL #SS$_NORMAL, R0 ; Return success  
00A5 331 RET  
00A6 332  
00A6 333 .END
```


LIB\$ADDX
Symbol table

- Add infinite precision integers C 8

15-SEP-1984 23:46:37 VAX/VMS Macro V04-00
6-SEP-1984 11:03:08 [LIBRTL.SRC]LIBADDX.MAR;1

Page 9
(4)

ADDEND	=	00000004		
AOVFL		00000050	R	01
ATEST		00000049	R	01
AUGEND	=	00000008		
DIFFERENCE	=	0000000C		
EXIT		0000004B	R	01
LENGTH	=	00000010		
LIB\$ADDX		00000000	RG	01
LIB\$SUBX		00000056	RG	01
LIB\$ INVARG		*****	X	00
MINUEND	=	00000004		
SS\$ INTOVF		*****	X	00
SS\$ NORMAL		*****	X	00
SUBTRAHEND	=	00000008		
SUM	=	0000000C		

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 (0.)	00 (0.)	NOPI	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
LIB\$CODE	000000A6 (166.)	01 (1.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG				

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.05	00:00:01.01
Command processing	117	00:00:00.35	00:00:02.30
Pass 1	72	00:00:00.48	00:00:04.92
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	72	00:00:00.40	00:00:02.57
Symbol table output	2	00:00:00.01	00:00:00.40
Psect synopsis output	3	00:00:00.00	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	297	00:00:01.30	00:00:11.29

The working set limit was 750 pages.
3941 bytes (8 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 15 non-local and 6 local symbols.
333 source lines were read in Pass 1, producing 14 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

LIB\$ADDX
VAX-11 Macro Run Statistics

- Add infinite precision integers^{D 8}

15-SEP-1984 23:46:37 VAX/VMS Macro V04-00
6-SEP-1984 11:03:08 [LIBRTL.SRC]LIBADDX.MAR;1

Page 10
(4)

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:LIBADDX/OBJ=OBJ\$:LIBADDX MSRC\$:LIBADDX/UPDATE=(ENHS:LIBADDX)

0203 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY